RESEARCH PAPERS OF WROCŁAW UNIVERSITY OF ECONOMICS No. 147
Advanced Information Technologies for Management – AITM 2010 2010

**Cyril Cassagnes, David Bromberg, Damien Magoni**
University of Bordeaux – LaBRI, Bordeaux, France
e-mail: {cyril.cassagnes; Bromberg; magoni}@labri.fr

# AN OVERLAY ARCHITECTURE FOR ACHIEVING TOTAL FLEXIBILITY IN INTERNET COMMUNICATIONS

**Abstract:** The current state of the art for Internet communications shows that they are not flexible enough considering the capabilities of the current equipments. The mobile equipments are still unable to transfer a communication from one device to another without interrupting the communication. In the same way, although we have the choice of many applications, it is still impossible to transfer a communication from one application to another in a given device without interrupting the communication. Only device mobility is currently ensured and in a very limited way. We have a wide access to layer 2 mobility with technologies such as WiFi, WiMax and 3 + G. We also have some limited access to layer 3 mobility with the Mobile IP architecture. This paper presents a new architecture designed to bring total flexibility to Internet communications.

**Keywords:** architecture, flexibility, overlay network.

## 1. Introduction

Internet communications are the backbones of the so-called global village. The information age would not exist without it. However, are today's communications flexible? The current Internet communications are still based on the paradigms set by the TCP/IP protocol stack 30 years ago. And they are lacking several key features. Although many efforts have been done recently to provide mobility, security and multicast, those efforts have been focused mainly on the equipments (e.g. computers, cell phones, etc.) themselves, not on the communication streams and users. In this paper we propose and describe a new architecture designed as an overlay that will execute on top of the TCP/IP protocol suite of the participating devices. We have called this architecture by the name CLOAK which means: Covering Layers Of Abstract Knowledge. This overlay architecture will add names and virtual addresses to devices and users that will enable total flexibility for all kinds of Internet communications. Flexibility means here the transparent handling of the breakdown and restore of transport layer connections (i.e. such as TCP connections).

The remainder of this paper is organized as follows. Section 2 outlines the previous work done on communication flexibility. Section 3 highlights the current limitations of Internet communications. In section 4, we discuss the aims of our architecture. Section 5 presents the design and features of our proposal. Section 6 describes the main components of our architecture. Section 7 presents some performance results obtained by simulations. Finally, we conclude the paper and present our future research work in section 8.

## 2. Related work

Of course many studies have already been carried out to solve the above issues. Concerning the management of the device mobility, the best known solution is Mobile IP [*IP Mobility*... 2002], a protocol standardized by the IETF which is a solution that was mainly inspired by the results of the Monarch project at Rice University. Mobile IP is an existing solution for enabling the movement of IP devices from one IP subnet to another without connection interruption. It does not enable the transfer of connections from one device to another. Defined in 1996, Mobile IP is still a proposed standard and its availability in the Internet is very scarce. Indeed it is useful mainly across operators but they are reluctant to deploy it as it requires active support from their routers and it can cause security issues. Mobile IP needs a router configured as a home agent on the device's original network and a router configured as a foreign agent on the device's visiting network. It requires signalling security between the agents and the device and the agents themselves. As configuring Mobile IP means, for a network operator, providing access to foreign customers, user authentication and accounting must also be setup. Mobile IP has also issues with firewalls and NATs. Finally, it has issues with triangle routing and binding updates.

Concerning the TCP connection management, we have TCP-Migrate [Snoeren et al. 2001; Snoeren, Balakrishnan 2000] developed at the MIT, that provides a unified framework to support address changes and disconnectivity. "Migrate allows legacy applications to adapt to today's highly mobile environment, and provides mobile-aware applications with a robust set of system primitives for disconnectivity support, resource conservation, and rapid re-instantiation of network connections. Migrate treats disconnection as a fundamental component of mobility, and enables applications to gracefully reduce their resource consumption during periods of disconnection and rapidly resume sessions upon reconnection". We have Rocks [Zandy, Miller 2002] developed at the University of Wisconsin that "protect sockets-based applications from network failures, such as link failures (e.g., unexpected modem disconnection), IP address changes (e.g., laptop movement, DHCP lease expiry) and extended periods of disconnection (e.g., laptop suspension)". We have Migratory TCP [Sultan et al. 2002] at Rutgers University which is "a transport layer protocol for building highly-available network services by means of transparent migration of the server endpoint of a live connection between cooperating servers that provide the same

service. The origin and destination server hosts cooperate by transferring supporting state in order to accommodate the migrating connection". Finally the Fault-Tolerant TCP [Zagorodnov et al. 2003; Bressoud et al. 2001] developed at the University of Texas "allows a faulty server to keep its TCP connections open until it either recovers or it is failed over to a backup. The failure and recovery of the server process are completely transparent to client processes connected with it via TCP". As already stated, all these projects only deal with TCP re-connection. They do not enable the total virtualization of a session. They do not enable to switch both applications and devices at your will. They are also all based on the domain name/IP address paradigm and do not provide the separation of naming and addressing planes.

Concerning the session management, SIP [Rosenberg et al. 2002] is a solution for managing applications at the session level. It does introduce user names in the form of e-mail like addresses but it does not use device or application identifiers such as CLOAK. As e-mail like addresses, the SIP names are still bound to specific network domains contrary to CLOAK whose entity identifiers are independent of the network architecture. The SIP naming scheme allows a setup call to be redirected but it does not allow switching devices or applications on the fly. Firewalls pose a particularly difficult challenge to SIP sessions. Because SIP can use TCP and a well known port, configuring a firewall to pass SIP is not too difficult. However, this does not help the media path which uses RTP over UDP on various ports and will be blocked by most firewalls. A firewall or a proxy that controls the firewall needs to understand SIP, be able to parse an INVITE request and a 200 OK response, extract the IP addresses and port numbers from the SDP, and open up "pin holes" in the firewall to allow this traffic to pass. The hole can then be closed when a BYE is sent or a session timer expires. NATs also cause serious problems for SIP. Because responses in SIP are routed using Via header fields, a device behind a NAT will stamp its non-routable private IP address in its Via header field of messages that it originates making this field useless. In general, IP addresses in a SIP message, such as Via and Contact header fields, or IP addresses in SDP message bodies are not rewritten by a NAT router and thus will not be routable. Another problem with NATs is the time span of the NAT address binding. For a UDP SIP session, the time period is determined by the application. If a binding were removed before a BYE was sent terminating the session, the connection would effectively be closed and future signalling impossible. We can see that advanced protocols (such as Mobile IP and SIP) as well as middleware proposals (such as TCP-Migrate or FT-TCP) are still too limited in their capabilities to provide total flexibility for Internet communications.

## 3. Current limitations of internet communications

### A. Issues

A communication network such as the Internet can basically be represented by a stack of three parts. The lower part of the stack is composed of the devices. These devices are

real telecommunication or networking equipments. They can be terminal equipments for use by end users such as computers, cell phones, etc., or intermediate equipments for use by network operators such as switches, routers, base stations, etc. The middle part of the stack is composed of the protocols, taking this word with a broad meaning. These protocols are implemented by a vast amount of various software that run on the above devices. They are defined in norms that the implementations respect in order to be able to interact with the others. Without the protocols no communication would be possible. The higher part of the stack is composed of the applications and their users. These applications are directly manipulated by the users in order to carry out the desired task and they use the underlying protocols to interact through the network. As we use them through a network, these applications are called network applications but depending on how they work, they can also be called distributed or *n*-tiers applications. The main issue with the Internet is that these three parts are tied together too strongly by the addresses and identifiers of the resources (i.e. devices and applications) and this impairs the flexibility of the communications.

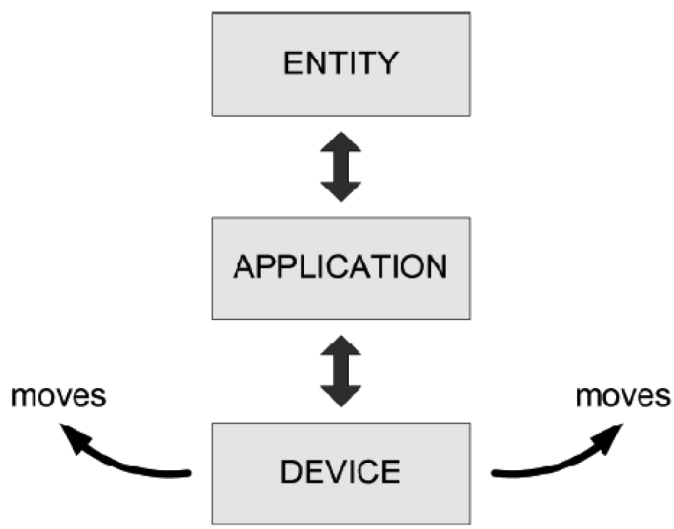


**Figure 1.** Current communication paradigm

Figure 1 shows the current communication paradigm of the Internet considering one communicating entity. An entity is using an application running on a device connected to a network. The device can move without interrupting the communication if it uses a wireless network interface and a mobility protocol such as 802.11 [ISO/IEC 2005] or HSPA [*3rd Generation*... 2008]. If the device is moving to another network operator (or at least to another IP subnet [*Internet Protocol* 1981]), an additional mobility protocol such as Mobile IP [*IP Mobility*... 2002] is required. If

the entity wants to switch to another device, the communication must be terminated. If the entity wants to switch to another application, the communication must also be terminated. If the entity is authenticated in the application, the communication cannot be transferred to another entity without breaking the communication. We can see that in the current state of the art, entities, applications and devices are tightly bound together (i.e. represented by arrows in Figure 1) during a communication and only the movement of the devices are currently supported. Note that on Figure 1 only one entity of a communication is shown, other entities would obey the same scheme.

**B. Causes**

For various historical reasons of architectural choices, Internet communications do not currently have the property of total flexibility. Here are some of the main reasons:

1. The IP address represents both a location and an identifier [*Internet Protocol* 1981]. In an Internet communication, the IP address depends on the location of the device (i.e. depends on the IP subnet it is attached to) and it also represents the unique identifier of the device. At the origin of the Internet, the computers were not able to be moved so this was not an issue. Today, devices move all around carried by their users or by vehicles and thus their location is changing all the time. On the other hand the IP address uniquely identifies the device in the Internet and serves as an anchor point for all its connections (see point 3). These two meanings of the IP address are orthogonal.

2. The port number represents sometimes both an application layer protocol identifier (for the server ports called well known ports and registered ports) and a multiplexing connection identifier. It is thus difficult to filter the network traffic as the port number does not always identify the type of traffic (see point 6). Furthermore a port number binds a given application with the connection in the operating system of the device. It is impossible to transfer a connection to another application without breaking it (see point 3). These two meanings of the port number are orthogonal.

3. In the Internet, any connection established by a transport layer protocol such as TCP [*Transmission Control Protocol* 1981] or UDP [*User Datagram Protocol* 1980] is bound to the IP addresses of each communicating device and to the port numbers of each communicating application. Thus it is not possible to change applications or to change devices during the communication without terminating the connection. Figure 2 shows the current layer architecture of the Internet (on an Ethernet network for example). The blocks represent the layers as defined in the OSI model, the rounded textboxes represent the protocol used at a given layer and the angled textboxes represent the programming interfaces used by code running at a given layer in the device. The arrows show that currently in the Internet, an application is using lower layer identifiers for establishing and maintaining a connection and this prevents the communication from being able to be switched to another application or another device on the fly.

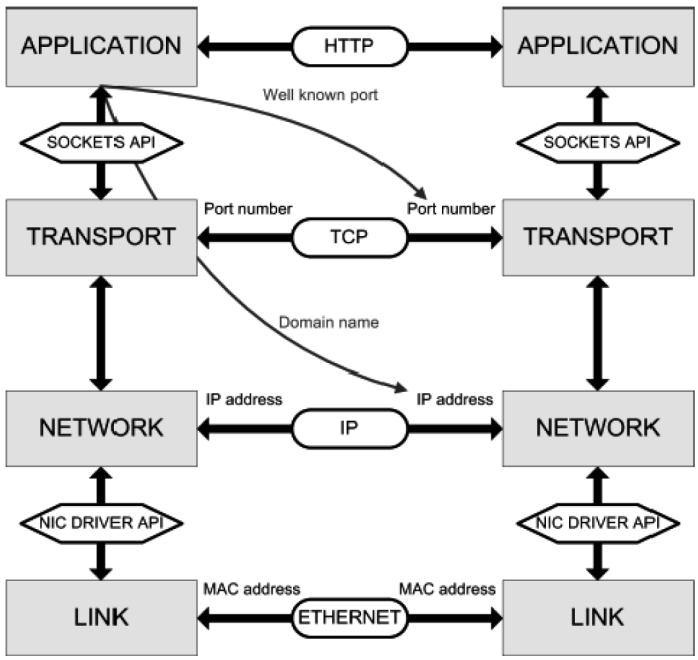


**Figure 2.** Current layer architecture of the Internet

4. The depletion of the IPv4 addressing space has led to the NAT [Srisuresh, Borch Egevang 2001] protocol. Unfortunately the NAT protocol is breaking the end-to-end model of the original Internet. That is, devices no longer have a unique public IP address but they are attributed a private IP address that is no longer unique except on the local network of the device (all NAT networks are using the same addressing spaces). Thus NAT devices become mandatory passage points that have to be taken into account. The Internet is not transparent from the source to the destination anymore and this is breaking the end-to-end principle of the original Internet architecture.

5. Entities are not clearly identified in the Internet. When there is a notion of user account, it is attached only to one specific application. The human entities, the users thus end up with dozens of different accounts ranging from social networking accounts to PayPal or bank accounts. When there is no notion of user account, an entity is most of the time identified by its IP address. If an application does not use an account, an Internet service will often rely on the IP address of the device that the entity is using in order to recognize the user. This lack of a common entity identifier used for a group of tasks and separated from any given device identifier leads to much management overhead.

6. The mainstream use of the Internet has attracted a lot of malicious users that have brought many security issues. One solution to these problems is the setup of

a firewall. A firewall is simply a packet filtering software running in a router or in the operating system of a device. A firewall inspects the headers of the incoming and outgoing packets and decides on the action to take with them (pass or drop). Of course filtering packets in the lower layers of the protocol stack severely restrain the flexibility of the communications.

7. In order to increase network security, specific protocols such as IPSec [Kent, Atkinson 1998] and TLS [Dierks, Rescorla 2008] are currently used in the Internet. However, any secure connection established by the IPSec protocol or by the SSL/TLS protocol is tied to the IP addresses of each communicating device by a security association. If one device moves in such a way that it changes its IP address or if one device changes, the secure connection as well as all connections above it are terminated.

## 4. Aims of our architecture

The aim of the CLOAK architecture and its corresponding protocols and software implementation is to bring total flexibility for all kinds of communications carried upon the Internet. At first, this aim seems outdated as everywhere we can read or hear about the ubiquitous and pervasive nature of the Internet and we are thus prone to conclude that flexibility is an inherent property of this communication network. However, a close look at the features of the current Internet applications in the next section will quickly show that we are still far from having total flexibility.

In the context of our architecture, a communication is a container of interactions between several entities. It need not be a duplex communication such as during a voice call, but it can be any form of simplex or duplex communication where information is processed and exchanged between the entities (e.g. talk, view video, check bank account, send mail, etc.). An interaction is simply a given type of action carried out between two or more entities by using an application protocol (e.g. FTP, HTTP, etc.). An entity is typically a human user but it can also be an automated service that we then call a server. A communication typically involves a minimum of two entities but it can involve many more in the case of multicast and broadcast communications. Finally, a device is a communication terminal equipment. On the device are running one or more hardwired or software programs called applications that are used by an entity to interact with other entities. Some devices (e.g. a telephone or a television set) support only one application while others (e.g. a computer or a 3G cell phone) can support many of them. A device is connected to a network operator. In the context of data communications, the network operator is itself connected to the Internet.

There are numerous communication networks but the CLOAK architecture focuses solely on the Internet. Each communication network is typically limited to one task and it is specifically engineered for it (radio network, telephone network, television network, etc.), however, since the advent of digitalized data, most people consider that a common digital communication network is the way to go as this

saves huge infrastructure costs. Internet is of course destined to be this common network. Not all communications currently pass through the Internet but already a large amount and this trend will continue until the other legacy networks fade out (actually their infrastructures may be used to support the Internet).

Given the above context, the definition of total flexibility is: *A communication has the property of total flexibility if it is able to be carried out without any definitive unwanted interruption when some or all of its components (i.e. device, application or entity) are evolving (i.e. moving or changing) over space and time.*

A communication thus has a lifetime that should be dependent only on the will of the currently implied entities. Changes in devices, applications and even entities (when it makes sense) should not terminate the communication.

**Examples of flexible Internet communications**

1. Mobile audio stream with transfer between devices A man is listening to the classical music channel of an Internet radio on the speakers of his desktop computer while working. At the end of the afternoon, the man transfers the connection on its 3G cell phone and continues to listen to it with his headphones while taking the bus back to his home. Arrived in the living room of his house, the man transfers again the connection on its high fidelity stereo amplifier in order to enjoy the music on his full range speakers. The man has been able to listen to his favourite symphony with no interruption between the movements and changes of devices.

2. Secure transaction with transfer between devices A woman is checking her bank account at home via the Internet. She is securely connected to one of the bank web servers. Later on, the server fails and the connection is transferred to another of the bank's server located in another town without any loss of information.

3. Mobile secure data connection A system administrator is working on a server from his desktop computer via a secure shell connection. He takes a company's car and moves from the central office to one of the company's offices located 50 km away. When he arrives on site, he connects to the Internet with his laptop and transfers his still alive shell session from his desktop to his laptop without any loss of information, his secure connection to the central office server is maintained.

4. Mobile video stream with transfer between entities and devices A journalist located in the Middle East is filming a war scene with a digital camera and he is moving around to closely follow the action. The video stream is directly sent via Internet to the journalist's correspondent located in the building of an Internet television company in Europe. The network connection on the battlefield is established through a mobile ad hoc network. After checking the video for a little while and assessing its importance, the correspondent decides to broadcast the live video to all the users watching the company's information channel. After a while, the camera battery level is getting low and the journalist decides to transfer its connection to a smaller backup camera that he is carrying. This transfer is done seamlessly with only a short interruption. All these above examples cannot be performed by using the current technologies available on the Internet. They require protocols and software that does not exist yet. We have seen why in the previous section.

## 5. Design and features of our architecture

### A. Design

In section 4 we have defined what we want to achieve in terms of functionalities and in section 3 we have explained why we do not yet have them with the current technologies. In this section we will explain how we can provide total flexibility to the Internet communications.

Figure 3 shows the CLOAK communication paradigm. In order to untie entities, applications and devices, CLOAK introduces the use of a session. A session is a communication descriptor that contains everything needed for linking entities, applications and devices together in a flexible way. A session represents the identity and the management information of a given communication. Thus the lifetime of a communication between several entities is equal to the lifetime of its corresponding session.

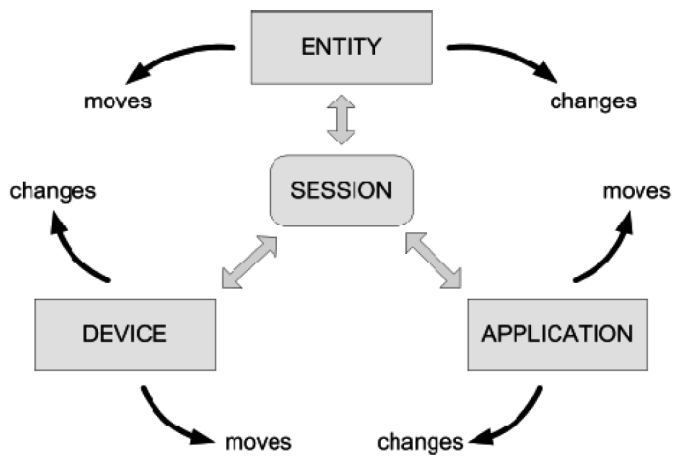


**Figure 3.** CLOAK communication paradigm

As shown on Figure 3, a device can move or be changed for another without terminating the session. Similarly, an application can be changed for another if deemed appropriate or even moved (i.e. mobile code) also without terminating the session. Finally entities can move or change (i.e. be transferred to another entity) without terminating the session if this is appropriate for a given communication. We can see that in our new architecture, entities, applications and devices are loosely bound together (i.e. represented by arrows on Figure 3) during a communication and all the movements and changes of devices, applications and entities are supported. Note that on Figure 3 only one entity of a communication is shown, other entities would obey the same scheme.

**B. Implementation**

To be able to implement our notion of session, we need to introduce several new layers of identifiers in order to decouple lower layers (devices) from middle layers (entities) and from higher layers (applications). Hence the name of this architecture. Please note that in this document we use the terms identifier and name for designing the same thing. We add naming layers above the existing ones in order to give abstract names to objects such as entities and devices. Thus we lay a cloak on the current Internet architecture. More specifically we need to define the following namespaces:

1. Device namespace: any device should be attributed a unique identifier that permanently represents the device. The lifespan of this identifier should be as long as the lifespan of its corresponding device.

2. Entity namespace: any entity should be attributed a unique identifier that represents the entity in a given context. It can be the name of a real person (John Smith) but it could also be the identifier of a professional function (Sales Manager) or the name of an organization (Michelin Company) or a specific service (Areva Accounting service). The lifespan of this identifier should be as long as the lifespan of its corresponding entity.

3. Application namespace: any application used during a part or all of a session should be attributed a unique identifier that enables it to receive data from the other applications of this session. The lifespan of this identifier should be equal to the lifespan of the use of the application. If the entity switches of application, this identifier becomes useless.

4. Session namespace: any session should be attributed a unique identifier that defines the session without doubt during its lifetime in the Internet. The lifespan of this identifier should thus last as long as the lifespan of the session itself.

Figure 4 shows the CLOAK layer architecture. CLOAK uses the session layer and the presentation layer between the transport and application layers. These layers do not exist in the Internet stack model but they do already exist in the seven layers OSI model. We add a CLOAK session protocol (CSP) at the session layer and a CLOAK interaction protocol (CIP) at the presentation layer. As shown in Figure 3, identifiers of devices, applications and entities are interwoven together inside a session, but for the purpose of implementation, we have to order them. We chose to manage a session and its involved devices at the session layer. We also chose to manage the interactions between entities at the presentation layer. An interaction is simply a given type of action carried out between two or more entities. This action is the use of an existing application layer protocol (e.g. FTP, HTTP, etc.). In order to carry out the chosen action, each entity chooses an application that knows how to speak the application protocol and that is suitable for both the device and entity using it. It must be noted that our architecture will of course use the existing application layer protocols as well as the existing transport layer protocols. Thus a file transfer (FTP [Postel, Reynolds 1985]) client application will still use the FTP protocol to

speak to a FTP server. Only the portion of code for establishing a session and thus a connection to the server will have to be rewritten for using the CLOAK API instead of the socket API [Wright, Stevens 1995]. The code implementing the application layer protocol will not have to be changed.

We have shown in Figure 4 how the CLOAK architecture will be implemented in the network protocol stack. We will show now how this implementation translates itself in the format of the data packets. Figure 5 shows the current headers of a typical packet exchanged between a web client and a web server. The application header involving the HTTP [Fielding et al. 1999] protocol is directly above the TCP protocol managing the connection in the operating system of the device.

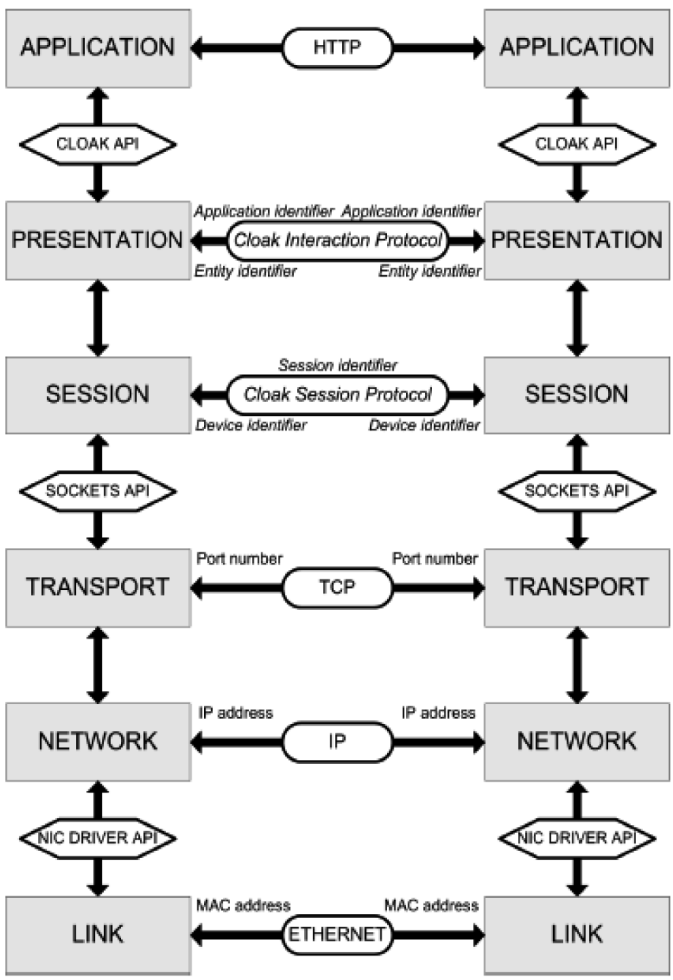


**Figure 4.** CLOAK layer architecture

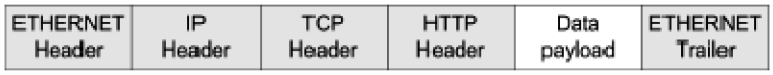


**Figure 5.** Example of a current protocol encapsulation

Figure 6 shows how would be a typical CLOAK packet exchanged between a web client and a web server. The application header involving the HTTP protocol is now located after the CLOAK headers. The CSP header is located directly above the TCP protocol managing the connection in the operating system of the device. It can be used to transfer the session to another device. The CIP header is located between the CSP and application level header. It is used for entity and application identification. It can serve for switching applications or entities when it makes sense in a communication.

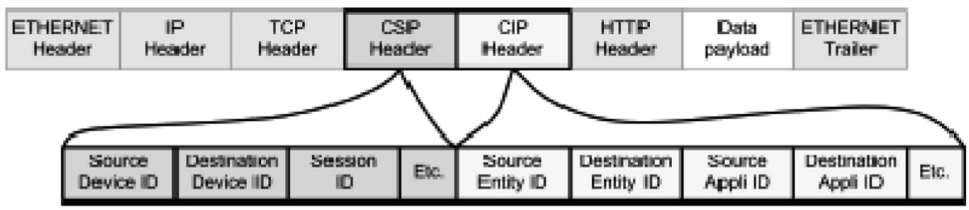


**Figure 6.** Example of a CLOAK protocol encapsulation

The definition and implementation of the CLOAK additional protocols (CSP and CIP) and their corresponding headers enable the architecture to solve the NAT issues because the applications using CLOAK will not use IP addresses and ports numbers for setting up or managing connections. They will use unique permanent entity identifiers, thus restoring the end-to-end principle of the Internet communications. The CLOAK architecture will also solve firewall issues because any kind and any number of transport level connections can be used by the applications using CLOAK. Thus on a given device, the applications can even use only a single port number and a single transport protocol if this is required by the firewall of the device. Indeed a CLOAK packet has a session ID field and two application ID fields that enable numerous applications to multiplex them on a single transport connection if necessary. CLOAK solves security issues because the security protocols can create security associations from entity IDs instead of IP addresses. The security is then by design independent from the devices and applications involved.

**C. Requirements**

In order to ensure that our architecture can be implemented and widely used in the Internet, it must follow several design requirements that we believe are mandatory for the success of its deployment:

1. Reliability: the CLOAK architecture should be mostly autonomous by considering the Internet as a dumb network providing end-to-end connections. Thus it should not rely on the use of intermediate components (proxies and middle boxes). It should also require no modifications in the network components (routers), no modifications in the operating system of the devices (terminal equipments) and no modification in the network services (e.g. Domain Name System). Only the applications that would like to use the CLOAK architecture should be slightly modified.

2. Scalability: the CLOAK P2P overlay should be able to remain efficient when it is leveraged by an important number of users, it should scale to at least dozens of thousands of users. That is why the CLOAK architecture for managing namespaces and sessions should be implemented as an overlay on top of the transport layer of the Internet. It should be a fully distributed, P2P system that would ideally be self-* (i.e. self-configuring, self-organizing and self-healing) as this will enable the incremental deployment of our architecture and ensure that the resources needed will be provided only by the devices using it.

Complexity: the CLOAK middleware should be able to be executed by most applications with a minimum number of modifications on the application side. The CLOAK architecture should have a low footprint on devices and should require minimal modifications in the code of the applications by having a simple and efficient API that should be easy to use for developers.

The architecture will be implemented as a middleware prototype that will enable devices to form a P2P overlay providing session flexibility for all its users.

## 6. Components of the architecture

### A. Multi-naming system

A multi-naming system hosts new namespaces at the session and presentation levels of the OSI model as well as its corresponding API. There will be a new namespace for the devices (containing records like: device ID – IP address), another for the entities (containing records like: entity ID – device ID), another for the applications (containing records like: application ID – session ID) and finally another for the sessions (containing records like: session ID – session data information). An application using CLOAK will not directly open a connection with an IP address and a port number as with the usual sockets API but it will only use the destination's entity ID as well as an interaction ID that will be put in a session having itself an ID. This multi-naming system will enable the separation advocated by the CLOAK architecture. The system will have to be fully dynamic (automatic updates) and distributed in order to be scalable. It can be implemented as a P2P overlay and be based on distributed hash tables. It should be autonomous, thus independent from any existing system (e.g. DNS) and should rely only on resources provided by collaborating devices. One crucial issue to solve for the system will be how to provide

and manage globally unique identifiers for all the above namespaces. The system will have to ensure the proper uniqueness of any defined identifier. Figure 7 shows a typical scenario relying on this naming system for solving an entity position.

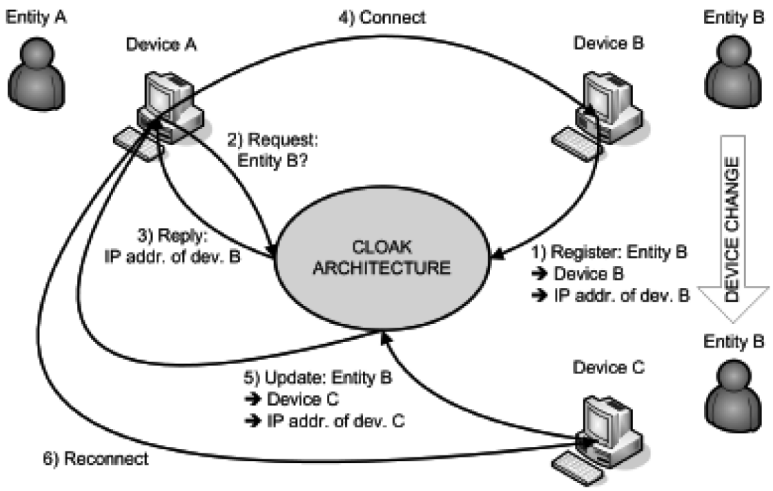


**Figure 7.** Identification and localization

The oval represents the distributed CLOAK architecture. An entity B registers itself in the system by providing the device ID it is on and its IP address. Any entity can retrieve the location of B by querying the system. It can then connect to B. Here a connection is just a transport layer connection between two devices, many can thus be created or destroyed many times during the life of a session. When B moves to another device during the same session, A can reconnect to B by being alerted by the system.

**B. Connection manager**

A system for managing CLOAK connections in the devices is required. A connection is an input/output endpoint for the end-to-end bidirectional transfer of data. A connection links two devices located anywhere in the Internet. The management system will be implemented between the applications and the operating system of the devices. It will provide a new API based on the use of entity names and interaction identifiers that will enable the creation of application level connections. The connection manager will transform application calls to system calls based on the use of IP addresses and port numbers (i.e. sockets). The connection manager will decouple the application from the operating system and thus from the device. This will enable the interruption of a system connection using a transport protocol (e.g. TCP, SCTP, etc.) and then the restoration of a new system connection with other parameters (e.g. new IP address, new transport protocol, new port number, etc.)

without interrupting the corresponding application connection. This capability will enable the movement of the device between various networks in a transparent way (without having to use Mobile IP) or the switch to another device during the same application connection. It will also allow, during the system reconnection, to change the transport protocol depending on the network conditions. The connection manager will also be able to transfer an application connection for an application to another, if this makes sense and if this is wanted by the user, also without cutting the connection. This connection manager will have to use the de facto socket API and will have to adapt to various operating systems. It will be implemented as a middleware running in the user space of the host operating system of the device.

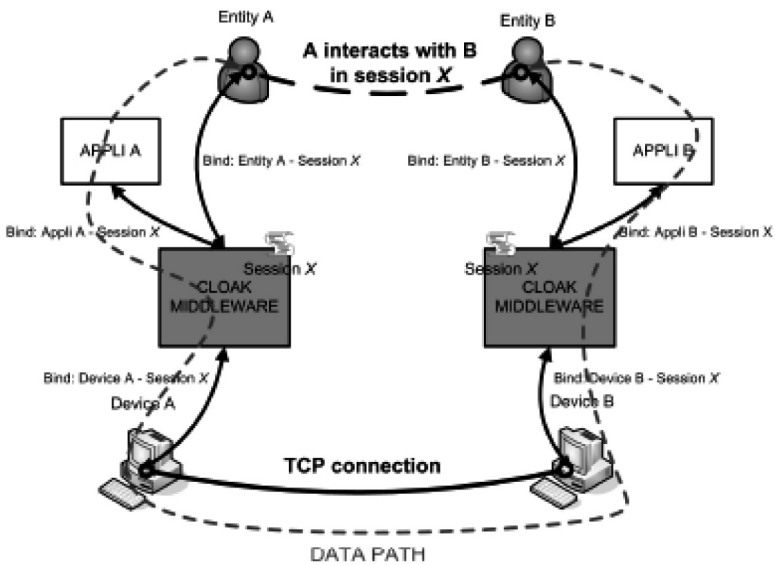


**Figure 8.** Connection management

Figure 8 shows a typical scenario relying on this connection management system. Two entities A and B are interacting in a session X. Let us assume that they are chatting. The connection manager in the devices are represented by gray boxes. The chat applications are represented by boxes. In the session X stored in the connection managers of both devices, can be found the bindings between the entity A, the application A and the device A and idem for B. The entity A is thus using the application A running on the device A to interact with the entity B using the application B running on the device B. The connection manager of A (here considering A as the communication initiator) is responsible for creating a transport level connection between the devices (which first requires the resolution of the device identifiers into IP addresses). The dotted line represents the data path between the

two chatting entities. For example, the data is generated by A that is typing some text in the application A. It is then transferred to the connection manager of the device A that forwards it into the proper TCP connection leading to the device B where the data will be forwarded to the application B and read by B.

**C. Session management system**

A system for managing CLOAK sessions is required. As defined above, a session is a communication's context container storing everything necessary to bind together entities, applications and devices that are involved in a given communication. Any device, application or entity can be changed or moved without terminating the session. In order to make this possible, the session should be stored in several devices (at least in all the communication's participating devices) and it should be easily transferable to other devices. Sessions will thus be virtual as they will not have to be stored and located on any specific device. This session management system will enable the survival of the session until all the entities involved decide to stop it. The system will have to be fully dynamic and distributed in order to be scalable. It can be implemented in a similar way as the multi-naming system. As such, it should be autonomous, thus independent from any existing system (e.g. SIP) and should rely only on resources provided by collaborating devices.

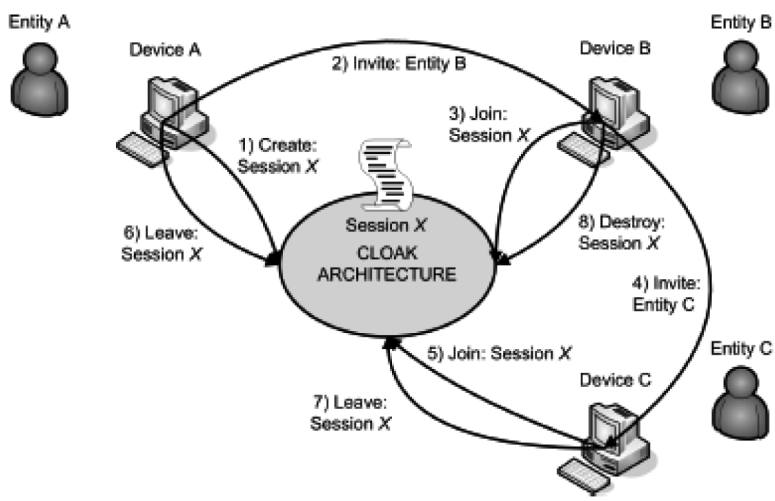


**Figure 9.** Session management

Figure 9 shows a typical scenario relying on this session management system. The oval represents the distributed CLOAK architecture. Let us assume that an entity A wants to start a video conference communication with an entity B. It first creates a session called X describing the desired interaction (e.g. video conference) as well as the destination entity that it wants to communicate with (here the entity B). Then

A sends an invite message to B that replies by joining the session X. Later on the entity B invites another entity C to participate in the video conference. C accepts and joins the session X. Three entities are now involved in the session X. Later on, the entity A leave the session X allowing the others to continue. This thus does not end the session X. Later on the entity C leaves the session X. The entity B being the last one involved decides to destroy the session and thus to end the communication.

## 7. Simulations

In this section we present some performance results of our overlay addressing and routing system obtained by simulations. Our overlay uses a hyperbolic tree addressing scheme and its corresponding greedy routing scheme as in [Kleinberg 2007]. We have used Internet maps created by real Internet data measurements (with *nec* [Magoni, Hoerdt 2005] and CAIDA [CAIDA 2010]. We have one IPv4 75k-node map from 2003, one BGP4 34k-node map from 2010 and one IPv6 4k-node from 2004. We use a precision of 10e-12 for all simulations.

In these simulations we have considered that every node in the maps is an overlay node and we have evaluated the hyperbolic addressing and routing system among them. The simulations are static because the nodes are always operational and the packets are instantly delivered between the nodes. We work with $d$-regular addressing trees and we carry out simulations to study the performance in function of this addressing tree degree. A randomly picked first node, called root, computes $d$ coordinates that it gives to its neighbours. Next, the $d$ children compute again $d$ coordinates. Each child repeats the process. Thus we try to address all the nodes of a map by a breadth first distribution algorithm. The simulations show that we cannot address all the nodes of the maps but the most part of it (typically above 90%). We evaluate the depth from the root, the distance, the stretch and the congestion on our three Internet maps: BGP4, IPv4 and IPv6.

We now look at the addressing distribution. In our hyperbolic addressing scheme each address has a fixed length of 16 bytes (2 8-byte double types), however the depth has a maximal value reached when the points are too close to the unit disk. This maximum depth depends on the degree chosen as well as the precision chosen. Here we study the influence of the chosen degree on the depth of the addressing tree.

Figure 10 shows the depth fluctuation in function of the tree degree. The degree ranges from 4 to 256. We see that in a hyperbolic tree, the increase of the degree has only a little influence on the tree depth for IP maps. However, for the BGP map, the depth of the addressing tree decreases when the tree becomes wider. Covering a map with tens of thousands of nodes only requires a depth around 25.

Now we study metrics obtained by the routing evaluation: the average distance, the stretch and the congestion. We also show here the influence of the degree as an input parameter.

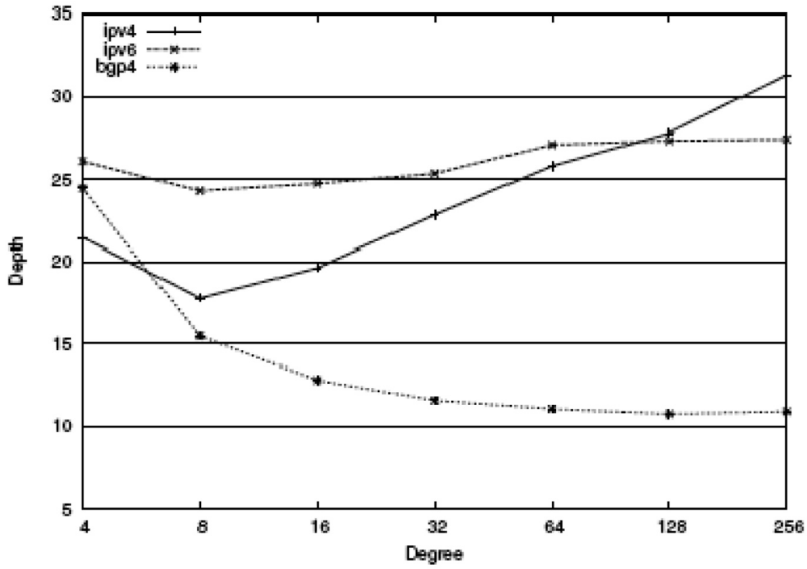


**Figure 10.** Maximum depth of the hyperbolic tree addressing scheme

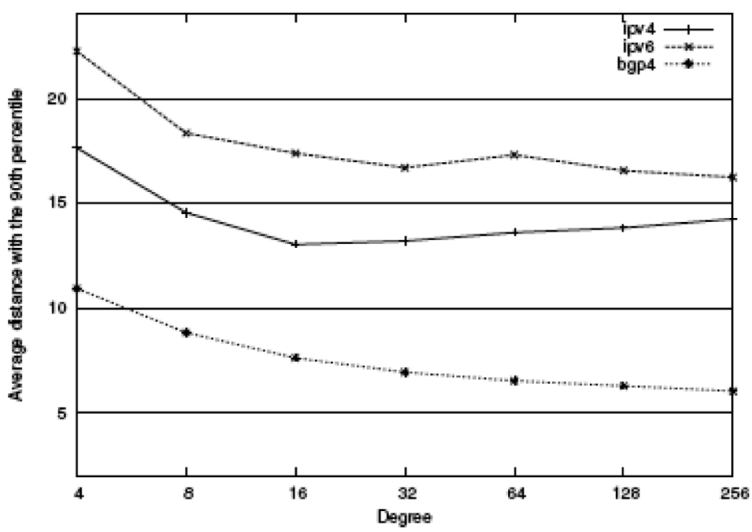


**Figure 11.** Average distance between nodes in the hyperbolic greedy routing scheme (90$^{th}$ percentile)

Figure 11 shows the average distance between nodes in the overlay. The distance is computed as the number of hops taken by the hyperbolic greedy routing. The smaller is the distance, the better is the routing efficiency. As we see, when the

degree is set higher, the distances are reduced. Moreover, we remark that BGP nodes have lowest distance which is expected as we measure AS hops and not IP hops here. The distance in the AS map are known to be shorter.

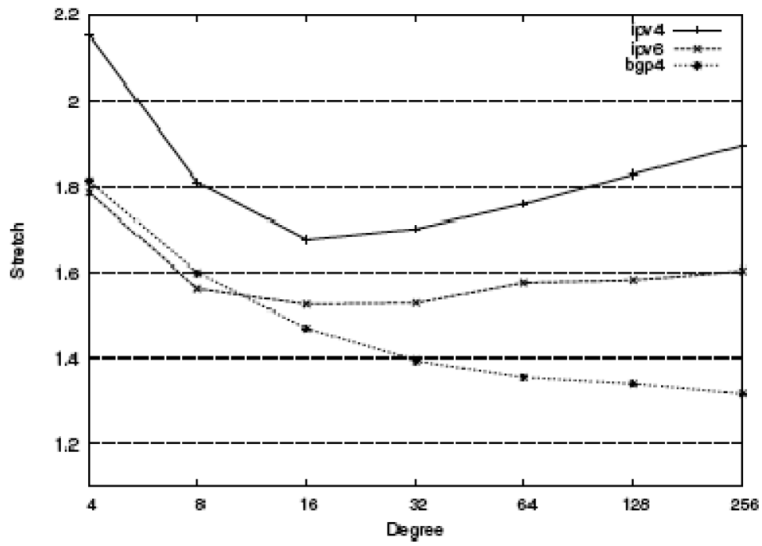


**Figure 12.** Average stretch in the hyperbolic greedy routing scheme

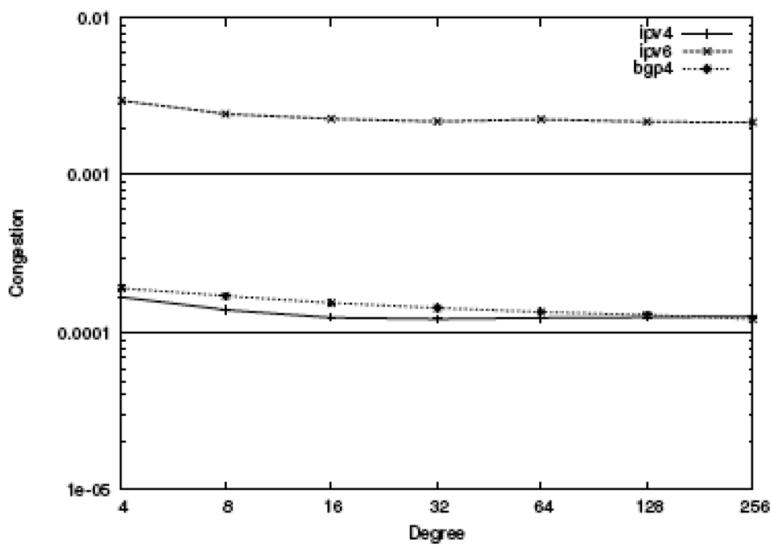


**Figure 13.** Congestion in the hyperbolic tree greedy routing scheme

In order to better evaluate the efficiency of this greedy routing scheme, we measure here the stretch of the routing paths. The stretch is equal to the hyperbolic greedy routing path length divided by the global routing shortest path length (i.e. the shortest possible path computed in a centralized way by the Dijkstra algorithm). Figure 12 shows that the degree has an impact on the stretch. The stretch is better when the degree increases although there is a diminishing return when the degree is above 32 for IP maps. The best stretch values for IP are respectively 1.7 with a degree of 16 and 1.5 with degree of 32. The value of the lowest stretch is 1.3 for the BGP map with a degree equal to 256.

Now we evaluate the efficiency of the hyperbolic greedy routing scheme by looking at the node congestion. We define the congestion of a node as the number of paths that went through it divided by the total number of paths tested. Figure 13 shows the congestion in the overlay (i.e. the average of the congestion of all nodes).

In this figure we observe that the congestion remains quite low. Furthermore, it is substantially the same between the IPv4 and BGP maps. However, we note that the IPv6 map has the highest congestion. This is because as this map is much smaller than the other two, the total amount of paths is much lower thus increasing the result. Finally, unlike the stretch, these plots show that the degree has a weak influence on the congestion.

## 8. Conclusions

In this paper we have presented a new architecture called CLOAK designed for providing flexibility to Internet communications. This architecture will be implemented as a middleware running on top of the TCP/IP stack of the devices and used by willing applications. The devices using the middleware will interconnect to each other and thus will form a P2P overlay network. This overlay will enable the applications to maintain their communications even if some transport layer connections are subject to failures. The middleware will transparently restore the transport connections without killing the applications. The architecture, by giving names to users and devices, will provide security and mobility to applications despite the IP address changes suffered by the devices.

We have already implemented the addressing and routing part of our middleware in a simulator and the preliminary results are encouraging. Our future work will be focused on the implementation of the multi-naming resolution system through the use of a scalable DHT mechanism and the test and evaluation of our middleware on a virtualized platform for studying the impact of transport layer connection pipelining created by the P2P overlay.

## References

*3rd Generation Partnership Project; Technical Specification Group Radio Access Network; High Speed Packet Access evolution; Frequency Division Duplex (Release 7)* (2008), http://www.quintillion.co.jp/3GPP/Specs/21902-700.pdf.

Bressoud T.C., El-Khashab A., Marzullo K., Zagorodnov D. (2001), Wrapping server-side TCP to mask connection failures, [in:] *Proceedings of the 20th Conference on Computer Communications*, IEEE, p. 329-338.

CAIDA (2010), *The Cooperative Association for Internet Data Analysis*, University of California, San Diego, http://www.caida.org.

Dierks T., Rescorla E. (2008), *The Transport Layer Security (TLS) Protocol Version 1.2*, RFC 5246, http://www.ietf.org/rfc/rfc5246.txt.

Fielding B., Gettys J., Mogul J., Frystyk H., Masinter L., Leach P., Berners-Lee T. (1999), *Hypertext Transfer Protocol - HTTP/1.1*, RFC 2616, http://www.ietf.org/rfc/rfc2616.txt.

*Internet Protocol* (1981), Ed. J. Postel, RFC 791, http://www.ietf.org/rfc/rfc791.txt.

*IP Mobility Support for IPv4* (2002), Ed. C. Perkins, RFC 3344, http://www.ietf.org/rfc/rfc3344.txt.

ISO/IEC 8802 11:2005(E) / IEEE Std 802.11i 2003 (2005), *Information technology – telecommunications and information exchange between systems – local and metropolitan area networks – specific requirements part 11: Wireless lan MAC and PHY specifications*, IEEE.

Kent S., Atkinson R. (1998), *Security Architecture for the Internet Protocol*, RFC 2401, http://www.ietf.org/rfc/rfc2401.txt.

Kleinberg R. (2007), Geographic routing using hyperbolic space, [in:] *Proceedings of the 26th Conference on Computer Communications*, IEEE, pp. 1902-1909.

Magoni D., Hoerdt M. (2005), Internet core topology mapping and analysis, *Computer Communications*, Vol. 28, Elsevier, pp. 494-506.

Postel J., Reynolds J. (1985), *File Transfer Protocol (FTP)*, RFC 959, http://www.ietf.org/rfc/rfc959.txt.

Rosenberg J., Schulzrinne H., Camarillo G., Johnston A., Peterson J., Sparks R., Handley M., Scooler E. (2002), *SIP: Session Initiation Protocol*, RFC 3261, http://www.ietf.org/rfc/rfc3261.txt.

Snoeren A.C., Balakrishnan H. (2000), An end-to-end approach to host mobility, [in:] *Proceedings of the 6th ACM MobiCom*, ACM, pp. 155-166.

Snoeren A.C., Balakrishnan H., Frans Kaashoek M. (2001), Reconsidering IP mobility, [in:] *Proceedings of the 8th HotOS*, IEEE, pp. 41-46.

Srisuresh P., Borch Egevang K. (2001), *Traditional IP Network Address Translator*, RFC 3022, http://www.ietf.org/rfc/rfc3022.txt.

Sultan F., Srinivasan K., Iyer D., Iftode L. (2002), Migratory TCP: Connection migration for service continuity in the Internet, [in:] *Proceedings of the 22nd International Conference on Distributed Computing Systems*, IEEE, pp. 469-470.

*Transmission Control Protocol* (1981), Ed. J. Postel, RFC 793, http://www.ietf.org/rfc/rfc793.txt.

*User Datagram Protocol* (1980), Ed. J. Postel, RFC 768, http://www.ietf.org/rfc/rfc768.txt.

Wright G., Stevens R. (1995), *TCP/IP Illustrated*, Vol. 2: *The Implementation*, Addison-Wesley, Reading, MA.

Zagorodnov D., Marzullo K., Alvisi L., Bressoud T. (2003), Engineering fault tolerant TCP/IP services using FT-TCP, [in:] *Proceedings of the International Conference on Dependable Systems and Networks*, IEEE, pp. 393-402.

Zandy V., Miller B. (2002), Reliable network connections, [in:] *Proceedings of the 8th ACM MobiCom*, ACM, pp. 95-106.

# REALIZACJA GLOBALNEJ ELASTYCZNOŚCI KOMUNIKACJI INTERNETOWEJ POPRZEZ ARCHITEKTURĘ NAKŁADKOWĄ

**Streszczenie:** Obecny stan prac badawczych nad komunikacją w Internecie wskazuje na brak dostatecznej elastyczności komunikacji spowodowany możliwościami obecnego sprzętu. Urządzenia mobilne mają wciąż trudności w przekazywaniu komunikatów z jednego urządzenia do drugiego bez przerywania połączenia. Podobna sytuacja jest z zastosowaniami, mimo wielu dostępnych programów nie jest jeszcze możliwe przekazanie komunikatu z jednej aplikacji do drugiej danego urządzenia bez przerywania połączenia. Jedynie mobilność urządzeń jest obecnie zapewniona, ale też w sposób bardzo ograniczony. Mamy szeroki dostęp do warstwy 2 w takich technologiach, jak WiFi, WiMax i 3 + G. Mamy również ograniczony dostęp do warstwy 3 mobilności z architekturą IP Mobile. W artykule przedstawiono nową architekturę, której zadaniem jest stworzenie możliwości globalnej elastyczności w komunikacji internetowej.